NOVA SCHOOL OF SCIENCE & TECHNOLOGY
DEPARTAMENTO DE INFORMÁTICA

Arquitetura de Computadores 2024/25

TPC 3

Entrega: 18h00 de 20de maio de 2025

Este trabalho de casa consiste num exercício de programação a ser realizado em grupo de no máximo dois alunos. Pode esclarecer dúvidas gerais com colegas, mas a solução e a escrita do código devem ser estritamente realizadas pelos membros do grupo. Os casos de plágio serão punidos de acordo com os regulamentos em vigor.

A forma de entrega será através do Mooshak, usando os identificadores do trabalho anterior, a menos que o grupo tenha sido alterado. Informação mais detalhada será enviada através de uma mensagem via CLIP.

Cifrar e decifrar com o algoritmo Tiny Encryption Algorithm (TEA)

Neste exercício implementam-se dois programas que permitem cifrar e decifrar um ficheiro usando o método TEA (Tiny Encryption Algoritm). O método TEA usa uma chave com 128 bits e o ficheiro é cifrado/decifrado dividindo-o em blocos com 64 bits. Os algoritmos para cifrar e decifrar um bloco são os seguintes:

- a chave está no vetor k com 4 inteiros de 32 bits
- os dados para cifrar/decifrar estão no vetor `text` com 2 inteiros de 32 bits

```
void encrypt(unsigned int text[], unsigned int k[]) {
  unsigned int y = text[0], z = text[1];
  unsigned int delta = 0x9e3779b9, sum = 0;
  int n;
  for (n= 0; n < 32; n++)
    {
      sum += delta;
      y += ((z << 4) + k[0]) ^ (z+sum) ^ ((z >> 5) + k[1]);
      z += ((y << 4) + k[2]) ^ (y+sum) ^ ((y >> 5) + k[3]);
    }
  text[0] = y;  text[1] = z;
}

void decrypt(unsigned int text[], unsigned int k[]) {
  unsigned int y = text[0], z = text[1];
  unsigned int delta = 0x9e3779b9, sum = delta << 5;
  int n;
  for (n= 0; n < 32; n++)
    {
      z -= ((y << 4) + k[2]) ^ (y + sum) ^ ((y >> 5) + k[3]);
      y -= ((z << 4) + k[0]) ^ (z + sum) ^ ((z >> 5) + k[1]);
      sum -= delta;
    }
  text[0] = y; text[1] = z;
}
```

O *Makefile* seguinte contém a forma de obter os três 3 ficheiros executáveis.

```
CC=gcc
CFLAGS=-Wall -std=c99 -g
ASFLAGS=-gstabs

all: genKey crypt-tea decrypt-tea

genKey: genKey.c.c
   $(CC) -Wall -g -o genKey genKey.c

cryptTEA: cryptTEA.c tea.o
   $(CC) $(CFLAGS) -o cryptTEA cryptTEA.c tea.o

decryptTEA: decryptTEA.c tea.o
   $(CC) $(CFLAGS) -o cryptTEA decryptTEA.c tea.o

tea.o:tea.s
   as $(ASFLAGS) -o tea.o tea.s

clean:
   rm -f *.o genKey cryptTEA decryptTEA
```

Segue-se a descrição dos 3 programas.

genKey

É invocado com:

```
./genkey file_name_to_store_key number_of_bytes_of_key
```

Obtém uma chave com o número de bytes especificado no 2º argumento; a chave é guardada em binário bits no ficheiro cujo nome é o 1º argumento. No caso do algoritmo TEA, a chave tem 16 bytes (128 bits).

cryptTEA

É invocado com:

```
./cryptTEA file_with_key  file_plain file_ciphered
```

Usa a chave que está no ficheiro cujo nome é o 1º argumento para codificar com o algoritmo TEA o ficheiro cujo nome é o 2º argumento; o ficheiro cifrado é guardado com o nome que é o 3º argumento.

decryptTEA

É invocado com:

```
/decryptTEA file_with_key  file_cyphered file_plain
```

Usa a chave que está no ficheiro cujo nome é o 1º argumento para descodificar com o algoritmo TEA o ficheiro cujo nome é o 2º argumento; o ficheiro com o conteúdo em claro é guardado com o nome que é o 3º argumento.

Trabalho a realizar

São fornecidos os ficheiros, *Makefile*, `genKey.c` e `decryptTEA.c` que não podem ser modificados. Devem ser desenvolvidos os ficheiros:

- `tea.s` - Onde estão implementadas, em assembly x86-64 sintaxe ATT, as funções `crypt` e `decrypt` acima especificadas . Os parâmetros de entrada são também os indicados.
- `cryptTEA.c` – Faça uma cópia do ficheiro `decryptTEA.c` para um ficheiro `cryptTEA.c`. Modifique-o de forma a que ele se comporte de acordo com a especificação acima. Uma alteração importante é que, para que o algoritmo funcione, o bloco de entrada tem de ter 64 bits (8 bytes); assim, se o ficheiro tiver um comprimento que não seja múltiplo de 8, é preciso acrescentar bytes a 0 no final, de forma a que o comprimento do ficheiro (em bytes) passe a ser múltiplo de 8. A esta operação chama-se *padding*.

Para testar a sua implementação basta, para vários tipos de ficheiros, executar:

```
./genKey key 16
./cryptTEA key xxx yyy
./decryptTEA key yyy zzz
diff xxx zzz
```

esperando que não haja diferenças. Note que, poderão ser detetadas diferenças se tiver havido *padding*.

Entrega

A entrega deve ser efetuada através do sistema Mooshak, como descrito no início; deve ser feita dentro do prazo, submetendo um ficheiro zip. Esse ficheiro deve conter apenas os ficheiros `cryptTEA.c` e `tea.s`.

Os programas serão compilados usando o ficheiro `Makefile` fornecido.